SC-T-213-VEFF - Web Programming I

Final Exam

Reykjavik University | Deadline: **See Canvas Assignment Deadline**

1 Overview

This is the final exam for T-213-VEFF, Web Programming I, in the Spring semester 2021. This exam is an *open-book* exam. Searching the internet, consulting books, and consulting course material is allowed. However, getting support using discussion forums (within the course or outside), chats, or direct interaction with other people (students or not) is plagiarism and leads to an immediate 0 (with potential other consequences, as regulated by the university). Similarly, directly copying material off the internet or other students is treated as plagiarism (according to the university's rules on studying and assessment. By participating in this exam, you consent to complete this task independently, without help from others.

For questions, we will primarily use Piazza. Post your questions as a private question to all instructors. During the exam, we will then answer questions two times: at 10:00, and at 11:00. Additionally, students with extra time get another slot at 12:00. We will take all private messages we got until then and answer them one after another. For instance, if you post a question at 9:50, you'll get an answer shortly after 10 (depending on how many questions we are answering before). If you post a question at 10:10, you'll probably have to wait until 11:00. We will not answer any questions that come in via mail or on Discord.

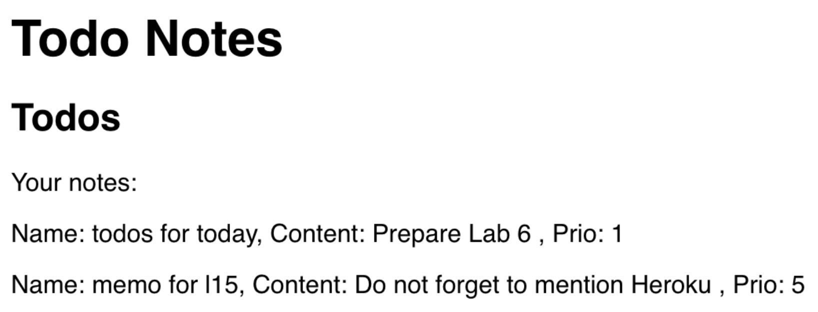
If you absolutely need to speak to an instructor, we will accommodate that via Zoom. In that case, also write a Piazza message but make clear that you'd like to speak to someone - we will then send you a Zoom link. Note that this might take longer than with written questions (just as you sometimes have to wait quite long in hall exams).

All answers have to be in English (by the university rules). We will disregard any Icelandic answers. Similarly, please write your questions on Piazza in English.

2 Task Overview

This exam consists of (a) a practical task spanning the content of all four course assignments, (b) a set of questions in which you briefly have to explain/describe your solution, and (c) a set of questions that require you to demonstrate broader knowledge of the course content. For (a), you are provided with a rudimentary application for handling todo notes, very similar to the one we used in the lectures. This application is found in the supplementary material to this assignment and contains a frontend and a backend. The frontend consists of an HTML, a CSS, and a JavaScript file. It is run by simply opening the HTML file in a browser. The backend is run by first executing npm install and then npm start in the backend folder. To execute backend tests, run npm test.

Loading the frontend while the backend is running should lead to two todo notes being displayed, as seen in Figure 1.


Todo Notes

Todos

Your notes:

Name: todos for today, Content: Prepare Lab 6 , Prio: 1

Name: memo for l15, Content: Do not forget to mention Heroku , Prio: 5

Figure 1: Frontend when backend is running

3 Practical Task (40 Points)

For the practical task, the following requirements apply:

1. It is allowed to change the port on which the backend is running.
2. It is allowed to change existing code (e.g., changing the data structure in the backend, changing the JS code in the frontend).
3. You are not allowed to introduce any additional dependencies (extra libraries or frameworks in the backend, extra npm dependencies).
4. You may leave the existing sample data in the backend, or add/change it to fit your purposes. However, your handed in solution needs to have some form of sample data that demonstrates its functionality.

By modifying the existing application, address the following tasks. Note that most of these tasks can be addressed in many possible ways.

Task 1 Instead of the current design, change the frontend so that each note is displayed as a 200px wide, grey box. The box shall contain the note name, content, and priority. Multiple boxes/notes should be displayed next to each other if the browser window is large enough. An example is displayed in Figure 2 for three notes and a window that is wide enough to fit two boxes, but not wide enough for three. Note that the figure also includes the delete buttons for Task 2. **(8 Points)**

Task 2 For each note, add a "delete button" in the frontend. When pressed, a DELETE request should be made to the backend to delete that note. If successful, the note should disappear from the frontend. You are allowed to redraw the notes afterwards, but not reload the entire page. **(8 Points)**

Task 3 In the backend, add a POST endpoint to allow adding a new note. The endpoint should check that the name, content, and priority variables are all supplied in the request, that name and content are not empty strings, and that the priority is a numerical value between 1 and 5 (including both). If successful, a new note is added with the provided values and an automatically-assigned numerical, unique id. The endpoint needs to follow the REST conventions used in the lectures and accompanying literature. **(8 Points)**

Task 4 For the backend endpoint that returns all notes, implement a filter by id range. When the query parameter named "greater_than" is supplied with a numerical value, only the notes with id larger than that value shall be returned. Other query parameters should be ignored. For instance, when a

GET request is made to (relative) URL */api/vEx0/notes?greater_than=15*, all notes with id 16 or larger are returned. **(8 Points)**

Task 5 Add a test to the existing test file *test/todosAllNPModified.test.js* that checks the success case for *DELETE /api/vEx0/notes/:noteId*. Assert that the status code is 200 for successfully deleting an existing note, and that the returned object is the note that was deleted (that it has the right values for all its attributes). **(8 Points)**

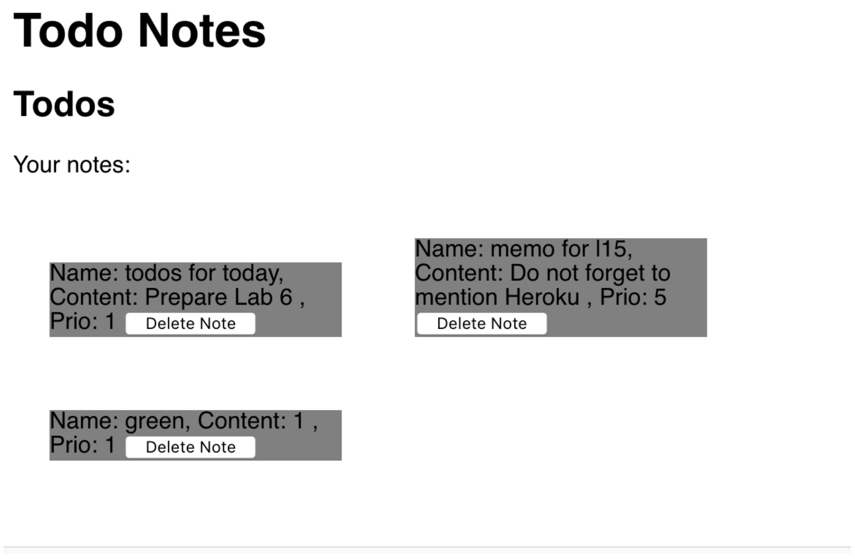

Todo Notes
Todos
Your notes:
Name: todos for today, Content: Prepare Lab 6 , Prio: 1
Delete Note
Name: memo for l15, Content: Do not forget to mention Heroku , Prio: 5
Delete Note
Name: green, Content: 1 , Prio: 1
Delete Note

Figure 2: Example of Solution for Task 1 and Task 2

4 Written Tasks

Answer the questions in the following two parts in written form. The answers should be placed in one text document which will be handed in as a PDF file.

4.1 Task Solution Explanations (20 Points)

For each Task in Section 3, explain briefly (in about 2-3 sentences) how you solved it. **(4 Points per task)**

4.2 Broader Questions (40 Points)

Q1 The frontend is currently not very accessible (in the sense accessibility was defined in the lectures). If you would be given the task to improve the accessibility in a short amount of time, what would be the most important aspects to focus on? Give arguments for your choice. **(8 Points)**

Q2 While the JavaScript code in the frontend is short, there are some parts that are cumbersome and repetitive. Which part of the JavaScript code (both the one provided and the one you wrote as a part of the practical part) should be replaced by calls to external libraries or frameworks? Give arguments for your choice. **(8 Points)**

Q3 The backend is RESTful in the sense discussed in the lectures. For one of the endpoints, give an example of how the endpoint could look like if it would violate the REST uniform interface constraint. Explain why such a violation would be of disadvantage compared to the RESTful solution. **(8 Points)**

Q4 The backend in its current form does not have any restrictions and could be susceptible to Denial of Service attacks, in which a large number of calls to the backend could overload the server. Without introducing authentication or authorization, how could the backend be improved to make such attacks less likely? Give arguments for your improvement suggestion. **(8 Points)**

Q5 The backend (including the added POST request) might be susceptible to XSS (Cross-Site Scripting) attacks. Discuss how an XSS scenario could look like in the case of this application. **(8 Points)**

Submission

The exam is submitted via Canvas. Submit a single zip file that contains all of the following files: (a) your modified backend and frontend, and (b) a PDF containing your textual answers. Do **NOT** include *node_modules* folders and/or *package-lock.json*.

Assessment

The practical task will be assessed on a code level. That is, points are awarded even if the project is not runnable (e.g., due to errors). For testing the JavaScript functionality and evaluating the design, we will use Chrome and Firefox.

Elegance of the written code is not an assessment criterion. However, if functionality runs excessive amounts of time (e.g., Task 3 taking a minute to execute successfully), or if the code is not understandable by the evaluators, points may be deducted.

For the written answers, it is evaluated whether the answers are factually correct, whether they actually answer the question, and whether the argumentation is consistent. If an answer contains many wrong parts, points may be deducted even if the correct answer is contained somewhere in the overall answer text.